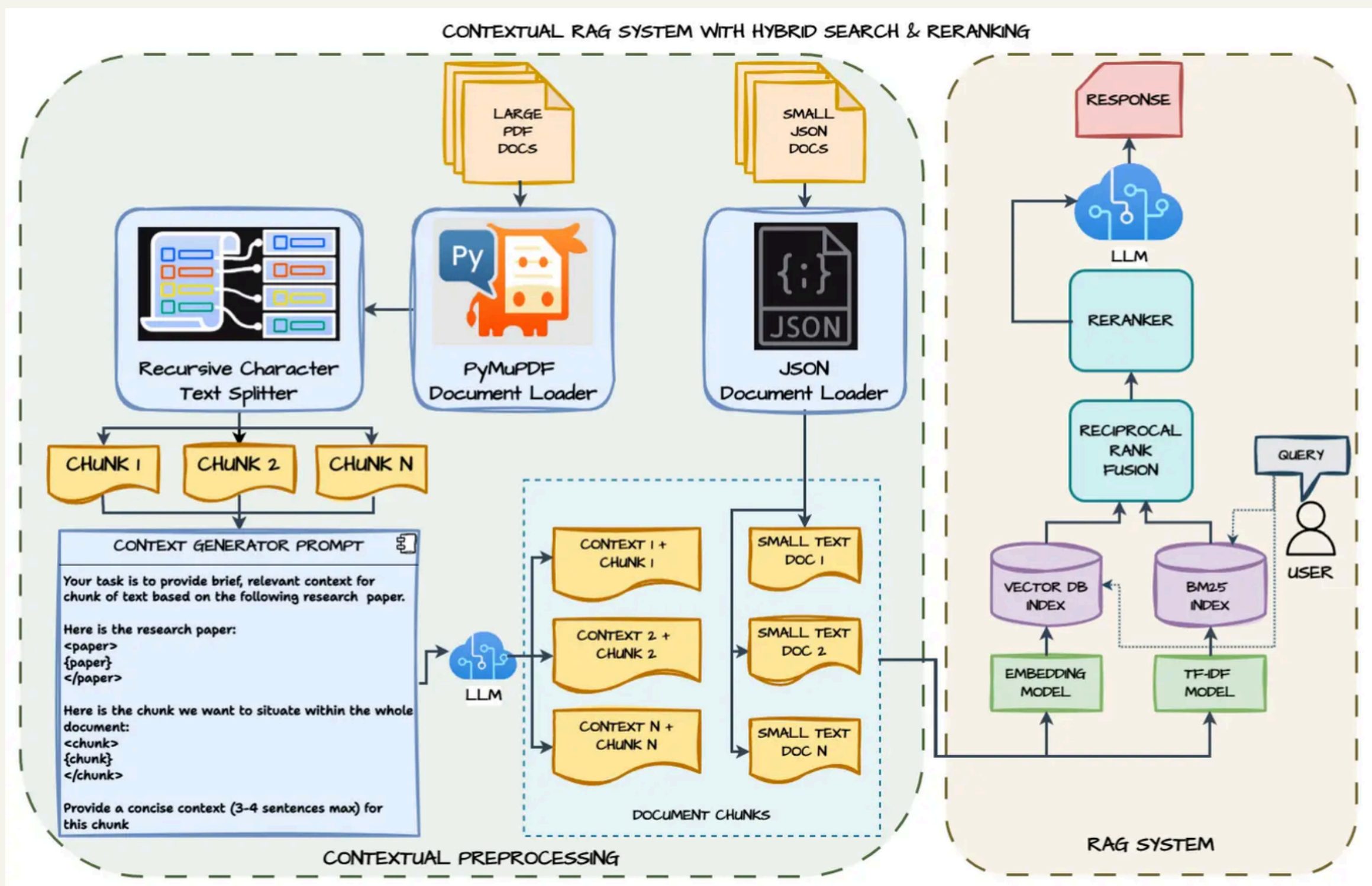




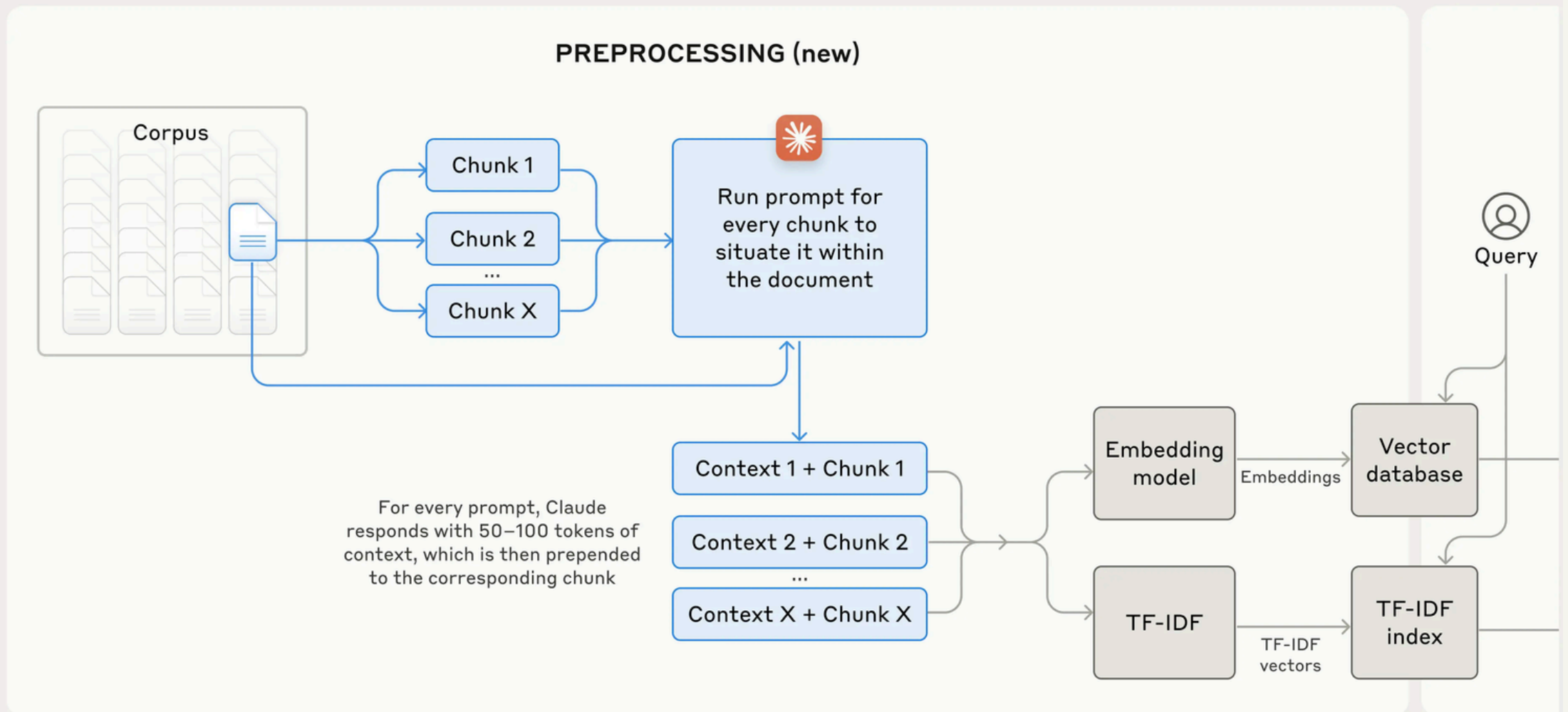
Improve RAG Performance with Contextual Retrieval

Hands-on Guide



Contextual Retrieval

Contextual Retrieval Preprocessing



- **Prepend chunk-specific explanatory context to each chunk before creating the vector DB embeddings and TF-IDF vectors**
- **Helps with having keywords or phrases in each chunk based on its relevant to the overall document**
- **Improves retrieval performance quite a bit which also helps with the overall RAG generation results because of better context**
- **The contextual chunking prompt can be built in various ways depending on your use-case**

Contextual Chunking

```
def generate_chunk_context(document, chunk):

    chunk_process_prompt = """You are an AI assistant specializing in research paper analysis.
    Your task is to provide brief, relevant context for a chunk of text
    based on the following research paper.

    Here is the research paper:
    <paper>
    {paper}
    </paper>

    Here is the chunk we want to situate within the whole document:
    <chunk>
    {chunk}
    </chunk>

    Provide a concise context (3-4 sentences max) for this chunk,
    considering the following guidelines:

    - Give a short succinct context to situate this chunk within the overall
    document for the purposes of improving search retrieval of the chunk.
    - Answer only with the succinct context and nothing else.
    - Context should be mentioned like 'Focuses on ....'
    do not mention 'this chunk or section focuses on...'

    Context:
    """

    prompt_template = ChatPromptTemplate.from_template(chunk_process_prompt)

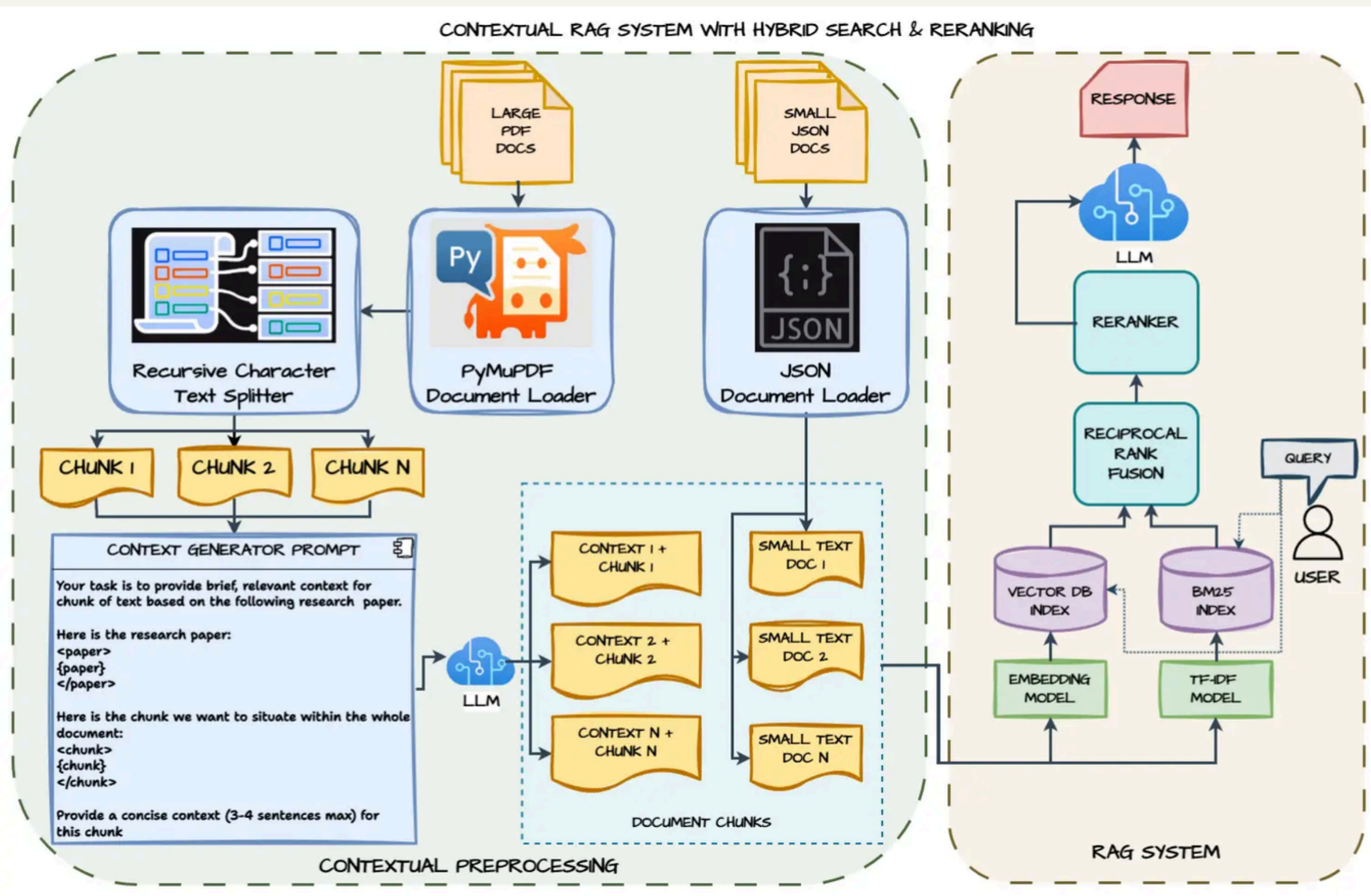
    agentic_chunk_chain = (prompt_template
                           |
                           llm
                           |
                           StrOutputParser())

    context = agentic_chunk_chain.invoke({'paper': document, 'chunk': chunk})

    return context
```


- Use a prompt recipe similar to the above example to create a short concise context for each chunk with respect to the overall paper
- Then prepend this context before the actual chunk content
- Pass it through the regular embedding and TF-IDF generation workflow as usual

Contextual RAG Architecture



- Perform initial hybrid retrieval to get the top potentially relevant chunks (Anthropic used the top 150)
- Pass the top-N chunks, along with the user's query, through the reranking model
- Using a reranking model, give each chunk a score based on its relevance and importance to the prompt, then select the top-K chunks (Anthropic used the top 20)
- Pass the top-K chunks into the LLM as context to generate the final response.

Article with Hands-on Code




DeepSeek Learning Paths GenAI Pinnacle Program Agentic AI Pioneer Program New

< Interview Prep Career GenAI Prompt Engg ChatGPT LLM Langchain RAG AI Agents Machine Learning Deep Learning GenAI Tools LLMOps Python NLP >

Home > RAG > Building Contextual RAG Systems with Hybrid Search and Reranking

Building Contextual RAG Systems with Hybrid Search and Reranking



Dipanjana (DJ) Sarkar

Last Updated : 12 Feb, 2025

22 min read

Download

234

Retrieval Augmented Generation systems, better known as RAG systems have become the de-facto standard to build Customized Intelligent AI Assistants answering questions on custom enterprise data without the hassles of expensive fine-tuning of [Large Language Models \(LLMs\)](#). One of the major challenges of naive RAG systems is getting the right retrieved context information to answer user queries. Chunking breaks down documents into smaller context pieces or chunks which can often end up losing the overall context information of the whole document. In this guide, we will discuss and build a Contextual RAG System inspired by [Anthropic's famous Contextual Retrieval approach](#) and couple it with Hybrid Search and Reranking using a complete step-by-step hands-on example. Let's get started!

New Feature Beta

Personalized GenAI Learning Path 2025 ✨
Crafted Just for YOU!

Download Now

Table of contents

1. Naive RAG System Architecture

2. Naive RAG System limitations

3. Standard Hybrid RAG Workflow

4. Understanding Contextual Retrieval

5. Implementing Contextual Retrieval

6. Contextual Retrieval Pre-Processing Architecture

7. Contextual RAG with Hybrid Search and Reranking Architecture

8. Hands-on Implementation of our Contextual RAG System

- Install Dependencies

ARTICLE LINK